REMARKS

Claims 2-8 are pending in the present application. The Office Action and cited references have been considered. Favorable reconsideration is respectfully requested.

Claims 2-8 were rejected under 35 U.S.C. §103 as being unpatentable over Baentsch (U.S. Patent No. 6,792,612) ("Baentsch '612") in view of Baentsch (U.S. Patent Application No. 2002/0093856) ("Baentsch '856"). These rejections are respectfully traversed for the following reasons.

Claim 7 recites a method for loading into a computer device with a programming language using objects, an updated release of an earlier application having earlier application classes and earlier static field identifiers, the updated release having updated application release classes, and updated static field identifiers, the programming language permitting an introduction of additional classes, a class hierarchy modification and a definition of further fields and methods, the method comprising the steps of computing, in a first computing operation prior to the loading, a class matching information establishing a correspondence between the earlier application release classes and the updated application release classes, computing, in a second computing operation prior to the loading, a second static field identifiers matching information establishing a correspondence between the earlier application release static field identifiers and the updated application release static field identifiers, linking the class matching information and the static field identifiers matching information to the updated application release as loaded into the computer device; and using the class matching information and the static field identifiers matching information to modify the objects to point at the updated application release classes and use the updated application release static field identifiers. This is not taught, disclosed or made obvious by the prior art of record.

With regard to claim 7, the Examiner cites Baentsch ‘612. Applicant respectfully disagrees.

The meaning of the word “object” is very specific in the case of programming languages using objects, or object-oriented programs. In the context of object-oriented programming, an object is a piece of data whose structure and functions are described in a class.

For the person ordinarily skilled in the art, in CAP files, classes cannot be referred to as objects, because they refer to classes (code) rather than data. Static fields cannot either by referred to as objects, because of their static nature.

Claim 7 mentions the loading of “an updated release of an earlier application”, in contrast with Baentsch ‘612’s “method for introducing new code”. The difference between the two concepts is explained in the figures below:

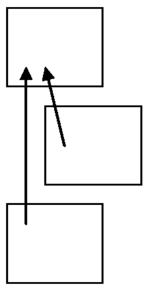
Figure 1

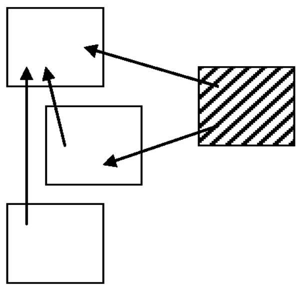
Figure 2

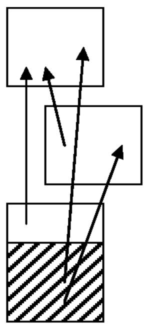
Figure 3

Figure 1 shows the state of the applications prior to the loading operation, with a few applications that are linked with each other.

Figure 2 shows the state of the applications after loading an application according to Baentsch ‘612‘s invention: new code has been introduced, and it is linked to the existing code. However, the existing code has not been modified in any way.

Figure 3 shows the state of the applications after loading an application according to the invention: one of the existing applications (an earlier application) has been modified by loading a new version of it (updated release).

“A new code” or “a new release” cannot be referred as “non-functional descriptive matter”. There is shown in the figures above a clear difference between the two, where Baentsch ‘612 “new code” is intended to be added to the system without modifying the already present code, whereas the new “new release”, according to the invention, is intended to “update” an “earlier application”, which will lead to the modification of already present code.

Of course, the loaded applications, or CAP files, contain the same categories of information in both inventions (classes, static field identifiers), as well as tables allowing the match or link between classes. However, the fundamental difference is that the invention establishes a correspondence “between earlier application release classes and updated application release classes”, whereas in Baentsch ‘612, fixup tables always establish the link between an existing class and a reference to that existing class in the currently loaded application.

Applicant respectfully submits that this difference is indeed significant, and that further, Baentsch ‘856 does not teach anything about updating applications. They only refer to the standard loading mechanisms for new applications.

More specifically, Baentsch ‘856 teaches in claim 1 that “said internal information is removed from said modified constant pool after linking”, which indicates that

Baentsch ‘856 makes no provision for further updates of the applications after its initial release.

With respect to claim 2, Applicant respectfully disagrees with the Examiner’s assertion. Lookup tables and fixup tables are similar, but their nature and use is different, as explained above with respect to claim 7.

With respect to claim 4, Baentsch ‘856 does not mention at any point the update of application data in [0008], which only covers the installation of application code (CAP file). Applicant respectfully submits that “procedures for updating application data”, differs from Baentsch ‘856‘s claim in at least two ways:

1. the procedures according to the invention aim at modifying the data of the application, and not only its code; and

2. then, the procedures according to the invention aim at updating the data, *i.e.*, at modifying existing data, rather than merely loading new data.

Further, such procedures would have been far from obvious to the person of ordinary skill in the art, at least because the data used in targeted systems (for instance, smart cards) are very sensitive, and modifying them requires specific procedures, as described in the invention.

Applicant respectfully submits that the purpose of the prior art methods is quite different from that of Applicant's method. For this reason, the teaching of Baentsch ‘612 and/or Baentsch ‘856 would not have been used by one of ordinary skill in the art to solve the problems faced by Applicant.

In view of the above amendment and remarks, Applicant respectfully requests reconsideration and withdrawal of the outstanding rejections of record. Applicant submits that the application is in condition for allowance and early notice to this effect is most earnestly solicited.

If the Examiner has any questions, he is invited to contact the undersigned at 202-628-5197.

Respectfully submitted,

BROWDY AND NEIMARK, P.L.L.C.
Attorneys for Applicant(s)

By /Ronni S. Jillions/
Ronni S. Jillions
Registration No. 31,979

RSJ:tdd
Telephone No.: (202) 628-5197
Facsimile No.: (202) 737-3528
G:\BN\M\Mout\Vetillard1\Pto\2009-01-21Amendment.doc